IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Steven W. Githens et al.	§	Filed: September 11, 2003
Serial No.: 10/660,143	§	Group Art Unit: 2175
Confirmation No.: 4972	§	Examiner: Jordany Nuñez

For: RICH GRAPHIC VISUALIZATION GENERATION FROM ABSTRACT DATA REPRESENTATION

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Jordany Nuñez, or electronically transmitted via EFS-Web, on the date shown below:

May 18, 2009 /Johnny Lam/
Date Johnny Lam

Dear Sir:

APPEAL BRIEF

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2175 dated December 18, 2008, finally rejecting claims 1-4 and 7-21. The final rejection of claims 1-4 and 7-21 is appealed. This Appeal Brief is believed to be timely since it is transmitted by the due date of May 18, 2009, as set by the filing of a Notice of Appeal on March 17, 2009.

Since an appeal brief fee in the amount of $510.00 had been paid for a previous appeal that did not reach a Board Decision, the fees due for filing this appeal brief is $30.00. The Commissioner is hereby authorized to charge $30.00 to counsel's Deposit Account No. 09-0465 / ROC920030276US1 for filing this appeal brief, and for any other fees required to make this appeal brief timely and acceptable to the Office.

TABLE OF CONTENTS

1.	Identification Page.....	1
2.	Table of Contents	2
3.	Real Party in Interest	3
4.	Related Appeals and Interferences	4
5.	Status of Claims	5
6.	Status of Amendments	6
7.	Summary of Claimed Subject Matter	7
8.	Grounds of Rejection to be Reviewed on Appeal	12
9.	Arguments	13
10.	Conclusion	23
11.	Claims Appendix	24
12.	Evidence Appendix	30
13.	Related Proceedings Appendix	31

Real Party in Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicant asserts that no other appeals or interferences are known to the Applicant, the Applicant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1-4 and 7-21 are pending in the application. Claims 1-47 were originally presented in the application. Claims 5-6 and 22-47 have been canceled without prejudice. Claims 1-4 and 7-21 stand finally rejected as discussed below. The final rejections of claims 1-4 and 7-21 are appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

All claim amendments have been entered by the Examiner. No amendments to the claims were proposed after the final rejection.

Summary of Claimed Subject Matter

Claimed embodiments include methods directed to generating a graphical representation of data.

A. CLAIM 1 – INDEPENDENT

Claim 1 recites a computer-implemented method of generating a graphical representation of data. *See* Application, page 3, lines 13-15. The method includes generating an abstract data structure defining a plurality of abstract attributes representing an abstract graphical representation of the data. *See id.* page 18, lines 8-16; page 23, lines 1-16; Figure 4A element 410; Figure 4B element 450. The method also includes providing transformation rules for transforming the abstract data structure into a concrete data structure. *See id.* page 28, lines 3-21; Figure 3 element 310. The transformation rules include a plurality of subsets of transformation rules, each subset describing graphical attributes of a requested graphical representation type and being specific to a different graphics rendering language, whereby the transformation rules support a plurality of graphical representation types and a plurality of graphics rendering languages. *See id.* page 30, lines 1-7; page 26, lines 24-27. The method also includes selecting a subset of the plurality of subsets of transformation rules in accordance with a requested graphical representation type. *See id.* page 28, lines 3-28. The method also includes generating, on the basis of the abstract data structure and the selected subset of transformation rules, a concrete data structure defining a concrete graphical representation of the data in a graphics rendering language, wherein generating the concrete data structure is done by operation of a computer processor. *See id.* page 28, lines 3-28; page 26, lines 28-29; page 27, lines 1-5; page 30, lines 8-10; Figure 3 element 314; Figure 4 element 470; Figure 5 element 570.

B. CLAIM 2 – DEPENDENT

Claim 2 depends from claim 1, which recites a computer-implemented method of generating a graphical representation of data. *See id.* page 3, lines 19-20. As claimed,

generating the abstract data structure includes determining the requested graphical representation type. *See id.* page 28, lines 5-21. Further, generating the abstract data structure includes selecting an abstract data structure template from the plurality of abstract data structure templates on the basis of the requested graphical representation type, the abstract data structure being generated using the selected abstract data structure template. *See id.* page 11, line 18 – page 12, line 2.

C. CLAIM 11 – INDEPENDENT

Claim 11 recites a computer-implemented method of generating a graphical representation of data. *See id.* page 3, lines 13-15. The method includes receiving a selection of a requested graphical representation type for a selected data set. *See id.* page 28, lines 5-21. The method also includes selecting an abstract data structure template from a plurality of abstract data structure templates, each being specific to a different graphical representation type and defining a plurality of template attributes for generically representing an abstract graphical representation in the respective different graphical representation type. *See id.* page 11, line 18 – page 12, line 2; page 30, lines 1-7; page 28, lines 3-28. Further, the selected abstract data structure template is specific to the selected graphical representation type. *See id.* page 26, lines 24-27. The method also includes generating, on the basis of the requested graphical representation type and the selected abstract data structure template, an abstract data structure defining a plurality of abstract attributes abstractly representing the data set in the graphical representation. *See id.* page 18, lines 8-16; page 23, lines 1-16; Figure 4A element 410; Figure 4B element 450. The method also includes providing transformation rules for transforming the abstract data structure into a concrete data structure, the transformation rules describing graphical attributes of the requested graphical representation type. *See id.* page 28, lines 3-21; Figure 3 element 310. The method also includes generating, on the basis of the abstract data structure, a concrete data structure defining a concrete graphical representation in a graphics rendering language using the transformation rules, wherein generating the concrete data structure is done by operation of a computer processor. *See id.* page 28, lines 3-28; page 26,

lines 28-29; page 27, lines 1-5; page 30, lines 8-10; Figure 3 element 314; Figure 4 element 470; Figure 5 element 570.

D. CLAIM 19 – INDEPENDENT

Claim 11 recites a computer-implemented method of generating an abstract data structure for a graphical representation of data. *See id.* page 3, lines 13-15. The method includes providing a plurality of abstract data structure templates, each abstract data structure template being associated with a specific graphical representation type. *See id.* page 11, line 18 – page 12, line 2; page 30, lines 1-7; page 28, lines 3-28. The method also includes determining a requested graphical representation type. *See id.* page 28, lines 5-21. The method also includes selecting an abstract data structure template from the plurality of abstract data structure templates on the basis of the requested graphical representation type. *See id.* page 28, lines 3-28. The method also includes generating an abstract data structure using the selected abstract data structure template. *See id.* page 23, lines 1-16; Figure 4B element 450. The method also includes transforming the abstract data structure into a plurality of concrete data structures, each concrete data structure corresponding to a different graphics rendering language, wherein transforming the abstract data structure is done by operation of a computer processor. *See id.* page 28, lines 3-28; page 26, lines 28-29; page 27, lines 1-5; page 30, lines 1-10; Figure 3 element 314; Figure 4 element 470; Figure 5 element 570.

E. CLAIM 20 – INDEPENDENT

Claim 20 recites a computer-implemented method of generating a graphical representation of data. *See id.* page 3, lines 13-15. The method includes receiving a selection of a graphical representation type for a selected data set. *See id.* page 28, lines 5-21. The method also includes selecting an abstract data structure template from a plurality of abstract data structure templates, each being specific to a different graphical representation type and defining a plurality of template attributes for generically representing an abstract graphical representation in the respective different graphical representation type. *See id.* page 11, line 18 – page 12, line 2; page 30, lines

1-7; page 28, lines 3-28. Further, the selected abstract data structure template is specific to the selected graphical representation type. *See id.* page 28, lines 3-28. The method also includes generating, on the basis of the selected abstract data structure template, an abstract data structure defining a logical representation of the data set graphically represented according to the selected graphical representation type. *See id.* page 23, lines 1-16; Figure 4B element 450. The method also includes transforming the abstract data structure into a plurality of concrete data structures, each concrete data structure corresponding to a different graphics rendering language, wherein transforming the abstract data structure is done by operation of a computer processor. *See id.* page 28, lines 3-28; page 26, lines 28-29; page 27, lines 1-5; page 30, lines 1-10; Figure 3 element 314; Figure 4 element 470; Figure 5 element 570.

F. CLAIM 21 – INDEPENDENT

Claim 21 recites a computer-implemented method of generating a graphical representation of data. *See id.* page 3, lines 13-15. The method includes receiving abstract attributes values comprising at least a selection of a requested graphical representation type for a selected data set. *See id.* page 28, lines 5-21. The method also includes selecting an abstract data structure template from a plurality of abstract data structure templates, each being specific to a different graphical representation type and defining a plurality of template attributes for generically representing an abstract graphical representation in the respective different graphical representation type. *See id.* page 11, line 18 – page 12, line 2; page 30, lines 1-7; page 28, lines 3-28. Further, the selected abstract data structure template is specific to the selected graphical representation type. *See id.* page 26, lines 24-27. The method also includes generating, on the basis of the received abstract attributes values and the selected abstract data structure template, an abstract data structure defining a plurality of abstract attributes abstractly representing the data set in the graphical representation. *See id.* page 18, lines 8-16; page 23, lines 1-16; Figure 4A element 410; Figure 4B element 450. The method also includes selecting transformation rules for transforming the abstract data structure into a concrete data structure from a plurality of transformation rules, the transformation rules describing graphical attributes of the

requested graphical representation type. *See id.* page 28, lines 3-21; Figure 3 element 310. The method also includes generating, on the basis of the abstract data structure, a concrete data structure defining a concrete graphical representation in a graphics rendering language using the transformation rules. *See id.* page 28, lines 3-28; page 26, lines 28-29; page 27, lines 1-5; page 30, lines 1-10; Figure 3 element 314; Figure 4 element 470; Figure 5 element 570.

Grounds of Rejection to be Reviewed on Appeal

1. Rejection of claims 1-4, 7-9, 11-17 and 19-21 under 35 U.S.C. § 103(a) as being unpatentable over *Cox et al.*, U.S. Pub. No. 20020156806 (hereinafter *Cox*) in view of *Baudel*, U.S. Pat. No. 6,928,436.

2. Rejection of claims 10 and 18 under 35 U.S.C. § 103(a) as being unpatentable over *Cox* in view of *Baudel*, further in view of *Koselj et al.*, U.S. Pat. No. 7,027,056 (hereinafter *Koselj*).

ARGUMENTS

1. Rejection of claims 1-4, 7-9, 11-17 and 19-21 under 35 U.S.C. § 103(a) as being unpatentable over *Cox* in view of *Baudel*.

2. Rejection of claims 10 and 18 under 35 U.S.C. § 103(a) as being unpatentable over *Cox* in view of *Baudel*, further in view of *Koselj*.

The Applicable Law

The Examiner bears the initial burden of establishing a prima facie case of obviousness. See MPEP § 2141. Establishing a prima facie case of obviousness begins with first resolving the factual inquiries of Graham v. John Deere Co. 383 U.S. 1 (1966). The factual inquiries are as follows:

(A) determining the scope and content of the prior art;

(B) ascertaining the differences between the claimed invention and the prior art;

(C) resolving the level of ordinary skill in the art; and

(D) considering any objective indicia of nonobviousness.

Once the Graham factual inquiries are resolved, the Examiner must determine whether the claimed invention would have been obvious to one of ordinary skill in the art.

Applicants' Response to the Examiner's Argument

Respectfully, Applicants submit that the Examiner has not properly characterized the teachings of the references and/or the claims at issue. Accordingly, a prima facie case of obviousness has not been established.

For example, the Examiner suggests that *Cox* discloses *transformation rules for transforming the abstract data structure into a concrete data structure . . . the transformation rules describing graphical attributes of a requested graphical representation type* as recited in claim 1. Specifically, the Examiner asserts as follows:

> Cox shows a computer implemented method of generating a graphical representation of data, comprising . . . retrieving and providing transformation rules for transforming the abstract data structure into a

> concrete data structure, the transformation rules comprising a plurality of subsets of transformation rules each subset describing graphical attributes of a requested graphical representation type (page 5, paragraph [0044], lines 12-23) . . . .

Final Office Action dated December 18, 2008, pages 2-3; *see also* Final Office Action dated September 25, 2007, page 3. The Examiner analogizes "the raw data being analyzed as the abstract data structure, and the implementation of the visualization… as the concrete data structure." *See, e.g.*, Final Office Action dated September 25, 2007, page 3. The Examiner also analogizes the "actions" in *Cox* to "transformation rules." *See id.* The "actions" are described in *Cox* as being selectable command options, presented to a user in "a list or through a graphical user interface," which "allow a user to change view parameters, select a subset of the author's data for viewing, or select specific data for display". *See Cox*, ¶ 44.

However, Applicants respectfully submit that the Examiner's analogy fails to explain how the cited material, as well as *Cox* generally, discloses the above limitation. For example, the Examiner suggests that "raw data" in *Cox* teaches an *abstract data structure*. However, Applicants claim an "abstract data structure defining a plurality of abstract attributes representing a graphical representation of data." The Examiner is apparently relying on "raw data" in Cox to teach both *data* and *abstract data structure defining a plurality of abstract attributes representing a graphical representation of data*. That is, by treating the *data* and the *abstract data structure defining a plurality of abstract attributes representing a graphical representation of data* as being identical, the Examiner is wholly ignoring substantive limitations in the claims (namely, "defining a plurality of abstract attributes representing a graphical representation of data"), thereby fundamentally misconstruing the claims. On this basis alone, the Examiner's rejection is defective and should be withdrawn.

In addition, the Examiner fails to explain how the "raw data" in Cox "defines a plurality of abstract attributes representing a graphical representation of . . . data." In fact, the "raw data" of *Cox* at best teaches *data*, and not *abstract data structure*. Further, Applicants claim "transformation rules for transforming the abstract data structure into a concrete data structure." For the reasons given above, *Cox* fails to teach "abstract data structure" as required by claim 1. Thus, it necessarily follows that

Cox also fails to teach "transformation rules for transforming the abstract data structure." On this basis alone, the Examiner's rejection is defective and should be withdrawn.

Further, even assuming, *arguendo*, that Cox somehow teaches "an abstract data structure defining a plurality of abstract attributes representing an abstract graphical representation of the data," the "actions" (i.e., a list of user command options) of *Cox* nevertheless fails to teach the recited *subsets of transformation rules. See* Advisory Action dated December 7, 2007 ("each action is a subset of a plurality of transformation rules (e.g., a view parameter may be changed, so that data displayed as a bar graph may be displayed as a pie chart) . . . ."). Respectfully, a user command option to "select specific data for display" or to "change a view parameter" is not analogous to a transformation rule. Nevertheless, the Examiner states as follows:

> [The] Examiner believes it is rather clear that when a user issues a command to transform a bar graph into a pie chart, graphical attributes of a requested graphical representation type are indeed described, for example, a graphical attribute of a pie chart, which is a graphical representation, is described and then displayed to the user.

Advisory Action dated February 20, 2009, page 2. In other words, the Examiner suggests that "a user command to display a pie chart" in *Cox* specifies graphical attributes of a requested graphical representation type (e.g., the pie chart). However, the user command to display a pie chart in *Cox* at best describes a requested graphical representation type (i.e., "to display a pie chart"), and not also *graphical attributes of the requested representation type* (e.g., graphical attributes of the pie chart). Further, the Examiner fails to explain how *Cox* teaches or suggests *graphical attributes of the requested representation type*. On this basis alone, the rejection is defective and should be withdrawn.

Further, even assuming, *arguendo*, that the "actions" of *Cox* teaches *graphical attributes of the requested representation type*, *Cox* nevertheless fails to teach the *subsets of transformation rules* as claimed. In regards to the *transformation rules*, the Examiner states:

> [The] Examiner believes it is rather clear that in order to display a bar graph as a pie chart a subset of transformation rules take place which command the displayed bar to be transformed into the pie chart. Thus, list

> of a user command options or actions is indeed a subset of transformation rules when one of said user command options or actions is to transform the bar graph into a pie chart.

Advisory Action dated February 20, 2009, page 2. Even assuming, *arguendo*, that "bar graph" and "pie chart" teach the recited *concrete data structure*, *Cox* at best teaches "transforming a first concrete data structure into a second concrete data structure." Such is not the same as the recited *transformation rules for transforming an abstract data structure into a concrete data structure*. In other words, by treating the *abstract data structure* and the *concrete data structure* as being identical, the Examiner is wholly ignoring substantive limitations in the claims, thereby fundamentally misconstruing the claims. Therefore, *Cox* fails to disclose the recited *transformation rules for transforming an abstract data structure into a concrete data structure*. On this basis alone, the rejection is defective and should be withdrawn.

Further, the Examiner suggests that *Cox* discloses the limitation of *abstract data structure templates, each . . . associated with a specific graphical representation type* as recited in claim 2. Specifically, the Examiner asserts as follows:

> Cox shows a computer implemented method of generating a graphical representation of data, comprising . . . providing and selecting an abstract data template (e.g., Bar Chart view object) from a plurality of abstract data structure templates (e.g., dynamic tables, text objects) (abstract) . . . .

Final Office Action dated December 18, 2008, pages 2-3; *see also* Final Office Action dated September 25, 2007, page 3. That is, the Examiner suggests that the recited *abstract data structure templates* are disclosed by, *e.g.*, a "BarChart view object." *Id.* Significantly, the Examiner's analogy fails to conform to the other limitations of claim 2. That is, claim 2 requires a limitation of *the abstract data structure being generated using the selected abstract data structure template.* The Examiner analogizes the "raw data being analyzed as the abstract data structure." *See* Final Office Action dated September 25, 2007, page 7. Respectfully, the Examiner's analogy leads to a contradictory result and is therefore untenable. That is, the Examiner's analogy requires the raw data ("abstract data structure") to be generated using a view object such as "BarChart" ("abstract data structure templates"). Such a requirement is wholly contradictory and is not disclosed by (or even consistent with) *Cox*. That is, *Cox* does not describe "raw data" as being generated with the "view objects." Therefore, contrary

to the Examiner's suggestion, *Cox* does not disclose *abstract data structure templates, each . . . associated with a specific graphical representation type*. Nevertheless, the Examiner states:

> Cox teaches Bar cart[*sic*] view object. One of ordinary skill in the art would readily understand the Bar Chart View object to be used to generate a Bar chart. Further, one of ordinary skill in the art would readily understand that raw data is needed to generate a bar chart, and that the bar chart view object uses the raw data to generate display instructions to a display device, said display device ultimately displaying the bar chart. Thus, Cox clearly teaches "the abstract data structure[''] (e.g., instructions to the display device) being generated using selected abastract[*sic*] data structure template (e.g., barch[*sic*] chart view object).

Advisory Action dated February 20, 2009, page 2. However, the Examiner's rationale fails to confirm to the other limitations of claim 1. For example, the Examiner suggests that the "abstract data structure" includes "instructions to the display device." However, Applicants claim a concrete data structure (rather than the abstract data structure, from which the concrete data structure is generated) that "[defines] a concrete graphical representation of the data in a graphics rendering language." In other words, by again treating the *abstract data structure* and the *concrete data structure* as being identical, the Examiner is fundamentally misconstruing the claims. Thus, the Examiner's rationale is improper for suggesting that *Cox* teaches *the abstract data structure being generated using the selected abstract data structure template* as required by claim 2. Therefore, *Cox* fails to disclose the limitation of *abstract data structure templates, each . . . associated with a specific graphical representation type* as recited in claim 2. Accordingly, Applicants submit that the rejection is defective and should be withdrawn.

Further, the Examiner suggests that *Cox* discloses *generating, on the basis of the abstract data structure . . . a concrete data structure defining a concrete graphical representation of the data in a graphics rendering language using the transformation rules* as recited in claim 1. Specifically, the Examiner states:

> Cox shows a computer implemented method of generating a graphical representation of data, comprising . . . generating, on the basis of the abstract data structure and the selected subset of transformation, a concrete data structure defining a concrete graphical representation in a graphics rendering language using the transformation rules; wherein generating the concrete data structure is done by operation of a computer processor (figure 8, and corresponding description) . . . .

Final Office Action dated December 18, 2008, page 3; *see also* Final Office Action dated September 25, 2007, page 3. That is, the Examiner analogizes a "live document display output" of *Cox*, Figure 8 to teach *generating . . . a concrete data structure* . . . . In other words, the Examiner's interpretation merely equates a "concrete data structure" with an "display output." Respectfully, the Examiner's interpretation trivializes the limitation of "generating . . . a concrete data structure defining a concrete graphical representation of the data in a graphics rendering language" as recited in claim 1. Although a display output may be a *result* of rendering according to a graphics rendering language, a display output does not define anything in any graphics rendering language.

Further, the description corresponding to Figure 8 in *Cox* (¶ 80-97) merely shows HTML code (i) for defining controls (¶ 83-87); and (ii) for invoking an applet (¶ 88-97). According to *Cox*:

> These controls are located within the text explaining their usage and in close proximity to the view bar chart view 832.
>
> Readers can also easily view the drivers with the most prize money by interacting with the "total winnings" histogram of view 834 or the related JavaScript controls . . . .
>
> . . . .
>
> ```
> <applet name=wins code=Idoc.BarApplet.class width=175
> height=130>
> <param name=url value="drivers.txt">
> <param name="Variable" value="Wins">
> </applet>
> ```

Cox, ¶ 80-81 and 94-97. As the cited passage illustrates, the controls in *Cox* are separate from the views in *Cox*, and are merely used to adjust the views. Therefore, following the Examiner's analogy (regarding the concrete data structure) yields a contradictory result, because the HTML code is not for generating a view (of the concrete data structure), but for defining a control, which is different from the view.

Further, the HTML code for invoking an applet merely invokes a Java applet (with two parameters: "url" and "Variable"). A Java applet is a software component that can run in a web browser. Moreover, the parameter "url value='drivers.txt'" is a reference to a text file containing raw data. In other words, the HTML code for invoking the applet is simply not any graphical representation of data. That is, the HTML code for invoking

the applet does not describe graphics at all. Merely *invoking a software component and supplying the software component with a filename of a file containing raw data* is not the same as a concrete graphical representation in a graphics rendering language. Further, even assuming, *arguendo*, that *Cox* somehow discloses *a concrete graphical representation in a graphics rendering language*, Cox nevertheless fails to disclose that the concrete graphical representation is generated using the transformation rules, as required by claim 1. Therefore, for the reasons set forth above, individually and collectively, *Cox* does not disclose *generating, on the basis of the abstract data structure . . . a concrete data structure defining a concrete graphical representation of the data in a graphics rendering language using the transformation rules*. Accordingly, Applicants respectfully submit that the rejection is defective and should be withdrawn.

Further, with respect to claim 1, the Examiner correctly acknowledges the following:

> Cox fails to specifically show: the transformation rules being specific to a different graphics rendering language, whereby the transformation rules support a plurality of graphics rendering languages; and transforming the abstract data structure into a plurality of concrete data structures, each concrete data structure corresponding to a different graphics rendering language.

See Final Office Action dated December 18, 2008, page 4 (emphasis added). Thus, the Examiner proposes to modify *Cox* with *Baudel* in order to teach the limitations at issue. Specifically, the Examiner asserts as follows:

> In the same field of invention, Baudel teaches: a method for graphically rendering information of a database. Baudel further teaches: a visualization of information stored in a database (col. 1, l. 7-13), a visualization of a data table being a program that given as input any instance of a data table, outputs a uniquely defined sequence of graphic language instructions (col. 3, l. 48-50), a graphic language being a set of programming language functions and data types that enable describing images on a computer screen, examples of which OpenGL, Poscript, Java3D (col. 3, lines 22-29), and a DECORATION process setting graphic attributes for each record being drawn, said attributes including certain illumination models described in languages such as OpenGL and Java3D (i.e., because this process sets graphics attributes in languages such as OpenGL and Java3D, the visualization described inherently may be output in those languages).

> Thus, it would have been obvious to one of ordinary skill in the art, having the teachings of Cox and Baudel at the time that the invention was made, to have combined the visualization of information stored in a database, a visualization of a data table being a program that given as input any instance of a data table, outputs a uniquely defined sequence of graphic language instructions, a graphic language being a set of programming language functions and data types that enable describing images on a computer screen, examples of which OpenGL, Poscript, Java3D, and a DECORATION process setting graphic attributes for each record being drawn, said attributes including certain illumination models described in languages such as OpenGL and Java3D of Baudel with the method as taught by Cox.

See Final Office Action dated December 18, 2008, page 4 (emphasis added). Further, the Examiner states:

> [O]ne of ordinary skill in the art would readily equate Baudel's teachings as follows: 1) the transformation rules (e.g., the transformation from a table of data to a visualization of a table, chart or graph based on said table of data) being specific (e.g., unique) to a different graphic rendering language (e.g., OpenGL, Poscript, Java3D), whereby the transformation rules support a plurality of graphics rendering languages; 2) and transforming the abstract data structure (e.g., table of data) into a plurality of concrete data structures (e.g., visualization of a table, chart, or graph based on said table of data), each concrete data structures (e.g., visualization) corresponding to a different graphics rendering language (e.g., a DECORATION process of said visualization would have unique illumination model attribute which depends to the graphic language used; thus using a different graphic language to produce the visualization would change the visualization.

Id., page 8 (emphasis added). Applicants respectfully submit that the Examiner is mischaracterizing the limitations at issue. For example, the Examiner suggests that *Baudel* teaches *transforming the abstract data structure into a plurality of concrete data structures*. In particular, the Examiner analogizes a "table of data" in *Baudel* to teach an *abstract data structure*. In other words, the Examiner's interpretation merely equates "abstract data structure" with "data." Respectfully, the Examiner's interpretation trivializes the limitation of "generating an *abstract data structure* defining a plurality of abstract attributes representing an abstract graphical representation of the *data*" as recited in claim 1. That is, *Baudel* does not disclose any abstract data structure defining a plurality of abstract attributes representing an abstract graphical representation of a table of data. Therefore, *Baudel* does not disclose *transforming the abstract data*

structure into a plurality of concrete data structures. Accordingly, Applicants respectfully submit that the rejection is defective and should be withdrawn.

Even assuming, *arguendo*, that *Baudel* somehow teaches *transforming the abstract data structure into a plurality of concrete data structures*, *Baudel* nevertheless fails to disclose *providing transformation rules for transforming the abstract data structure into a concrete data structure, the transformation rules comprising a plurality of subsets of transformation rules each subset describing graphical attributes of a requested graphical representation type and being specific to a different graphics rendering language*, as recited in claim 1. The Examiner suggests the contrary. Specifically, the Examiner states:

> Baudel explicitly teaches transformation rules (e.g., the rules used to transform a table of data into a visualization of said data), transforming the abstract data structure (e.g., table of data) into a concrete data structure (e.g., a visualization).

Final Office Action dated December 18, 2008, page 8. Curiously, the Examiner does not provide any citation for the "explicit" teaching. In fact, *Baudel* in its entirety fails to disclose *transformation rules* of any sort. In other words, the Examiner is merely inferring a teaching of "transformation rules" from "creating a visualization of a table from a table of data using graphic language instructions" in *Baudel*. However, Applicants claim "transformation rules . . . describing graphical attributes of a requested graphical representation type and being specific to a different graphics rendering language." Significantly, *Baudel* fails to disclose any transformation rules that describe graphical attributes of a requested graphical representation type or that are specific to a different graphics rendering language. Therefore, *Baudel* does not disclose *providing transformation rules for transforming the abstract data structure into a concrete data structure, the transformation rules comprising a plurality of subsets of transformation rules each subset describing graphical attributes of a requested graphical representation type and being specific to a different graphics rendering language* as recited in claim 1. Accordingly, Applicants respectfully submit that the rejection is defective and should be withdrawn.

Further, the Examiner's rationale for the proposed modification/combination is merely conclusory. "[R]ejections on obviousness grounds cannot be sustained by mere

conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness". *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006); cited with approval in *KSR Int'l Co. v. Teleflex Inc.*, 127 S.Ct. 1727, 1740-41 (2007). For example, regarding the proposed modification of *Cox* with *Baudel* the Examiner states:

> [I]t would have been obvious to one of ordinary skill in the art, having the teachings of Cox and Baudel at the time that the invention was made, to have combined the visualization of information stored in a database, a visualization of a data table being a program that given as input any instance of a data table, outputs a uniquely defined sequence of graphic language instructions, a graphic language being a set of programming language functions and data types that enable describing images on a computer screen, examples of which OpenGL, Poscript[*sic*], Java3D, and a DECORATION process setting graphic attributes for each record being drawn, said attributes including certain illumination models described in languages such as OpenGL and Java3D of Baudel with the method as taught by Cox.

Final Office Action, page 4. In other words, the Examiner merely restates benefits of the references (e.g., "visualization of information"). With all due respect, it is not clear how the benefits are achieved or enhanced by virtue of the proposed modification. More fundamentally, the Examiner simply has not demonstrated how the references will be combined/modified. Instead, the Examiner simply posits that the references can be combined/modified on the basis of generally desirable features, such as visualization, without any specific explanation of how the references would synergistically interoperate to produce the claimed invention. Accordingly, the rejection is defective and should be withdrawn.

Therefore, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

CONCLUSION

The Examiner errs in finding that:

1. Claims 1-4, 7-9, 11-17 and 19-21 are unpatentable over *Cox* in view of *Baudel*; and
2. Claims 10 and 18 are unpatentable over *Cox* in view of *Baudel*, further in view of *Koselj*.

Withdrawal of the rejections and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. MCCLELLAN, Reg. #44227/

Gero G. McClellan
Registration No. 44,227
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellant(s)

CLAIMS APPENDIX

1. (Previously Presented) A computer-implemented method of generating a graphical representation of data, comprising:

generating an abstract data structure defining a plurality of abstract attributes representing an abstract graphical representation of the data;

providing transformation rules for transforming the abstract data structure into a concrete data structure, the transformation rules comprising a plurality of subsets of transformation rules each subset describing graphical attributes of a requested graphical representation type and being specific to a different graphics rendering language, whereby the transformation rules support a plurality of graphical representation types and a plurality of graphics rendering languages;

selecting a subset of the plurality of subsets of transformation rules in accordance with a requested graphical representation type; and

generating, on the basis of the abstract data structure and the selected subset of transformation rules, a concrete data structure defining a concrete graphical representation of the data in a graphics rendering language; wherein generating the concrete data structure is done by operation of a computer processor.

2. (Previously Presented) The method of claim 1, wherein generating the abstract data structure comprises:

providing a plurality of abstract data structure templates, each abstract data structure template being associated with a specific graphical representation type;

determining the requested graphical representation type; and

selecting an abstract data structure template from the plurality of abstract data structure templates on the basis of the requested graphical representation type;

the abstract data structure being generated using the selected abstract data structure template.

3. (Original) The method of claim 2, wherein the requested graphical representation type is one of a bar chart, a line chart, a pie chart, a scatter plot and a combination thereof.

4. (Original) The method of claim 2, wherein the plurality of abstract data structure templates is associated with a particular data source of the data.

5. (Canceled)

6. (Canceled)

7. (Previously Presented) The method of claim 1, wherein the requested graphical representation type is one of a bar chart, a line chart, a pie chart, a scatter plot and a combination thereof.

8. (Original) The method of claim 1, wherein at least one of the abstract data structure and the concrete data structure is defined in Extensible Markup Language (XML).

9. (Original) The method of claim 1, wherein the concrete data structure is defined in a vector-based graphics language.

10. (Original) The method of claim 9, wherein the vector-based graphics language is one of Vector Markup Language (VML), Scalable Vector Graphics (SVG), and Hypertext Markup Language (HTML) Image Maps.

11. (Previously Presented) A computer-implemented method of generating a graphical representation of data, comprising:

receiving a selection of a requested graphical representation type for a selected data set;

selecting an abstract data structure template from a plurality of abstract data structure templates, each being specific to a different graphical representation type and defining a plurality of template attributes for generically representing an abstract graphical representation in the respective different graphical representation type, wherein the selected abstract data structure template is specific to the selected graphical representation type;

generating, on the basis of the requested graphical representation type and the selected abstract data structure template, an abstract data structure defining a plurality of abstract attributes abstractly representing the data set in the graphical representation;

providing transformation rules for transforming the abstract data structure into a concrete data structure, the transformation rules describing graphical attributes of the requested graphical representation type; and

generating, on the basis of the abstract data structure, a concrete data structure defining a concrete graphical representation in a graphics rendering language using the transformation rules; wherein generating the concrete data structure is done by operation of a computer processor.

12. (Original) The method of claim 11, further comprising:

rendering the data set, as described in the graphics rendering language, in a graphic.

13. (Original) The method of claim 11, wherein the graphical representation type is one of a bar chart, a line chart, a pie chart, a scatter plot and a combination thereof.

14. (Original) The method of claim 11, wherein the plurality of abstract data structure templates is associated with a particular data source of the data.

15. (Original) The method of claim 11, further comprising:

selecting a subset of the transformation rules in accordance with the graphical representation type; and

generating the concrete data structure using the subset of the transformation rules.

16. (Original) The method of claim 11, wherein at least one of the abstract data structure and the concrete data structure is defined in Extensible Markup Language (XML).

17. (Original) The method of claim 11, wherein the concrete data structure is defined in a vector-based graphics language.

18. (Original) The method of claim 17, wherein the vector-based graphics language is one of Vector Markup Language (VML), Scalable Vector Graphics (SVG), and Hypertext Markup Language (HTML) Image Maps.

19. (Previously Presented) A computer-implemented method of generating an abstract data structure for a graphical representation of data, comprising:

providing a plurality of abstract data structure templates, each abstract data structure template being associated with a specific graphical representation type;

determining a requested graphical representation type;

selecting an abstract data structure template from the plurality of abstract data structure templates on the basis of the requested graphical representation type;

generating an abstract data structure using the selected abstract data structure template; and

transforming the abstract data structure into a plurality of concrete data structures, each concrete data structure corresponding to a different graphics rendering language; wherein transforming the abstract data structure is done by operation of a computer processor.

20. (Previously Presented) A computer-implemented method of generating a graphical representation of data, comprising:

receiving a selection of a graphical representation type for a selected data set;

selecting an abstract data structure template from a plurality of abstract data structure templates, each being specific to a different graphical representation type and defining a plurality of template attributes for generically representing an abstract graphical representation in the respective different graphical representation type, wherein the selected abstract data structure template is specific to the selected graphical representation type;

generating, on the basis of the selected abstract data structure template, an abstract data structure defining a logical representation of the data set graphically represented according to the selected graphical representation type; and

transforming the abstract data structure into a plurality of concrete data structures, each concrete data structure corresponding to a different graphics rendering language; wherein transforming the abstract data structure is done by operation of a computer processor.

21. (Previously Presented) A computer-implemented method of generating a graphical representation of data, comprising:

receiving abstract attributes values comprising at least a selection of a requested graphical representation type for a selected data set;

selecting an abstract data structure template from a plurality of abstract data structure templates, each being specific to a different graphical representation type and defining a plurality of template attributes for generically representing an abstract graphical representation in the respective different graphical representation type, wherein the selected abstract data structure template is specific to the selected graphical representation type;

generating, on the basis of the received abstract attributes values and the selected abstract data structure template, an abstract data structure defining a plurality of abstract attributes abstractly representing the data set in the graphical representation;

selecting transformation rules for transforming the abstract data structure into a concrete data structure from a plurality of transformation rules, the transformation rules describing graphical attributes of the requested graphical representation type; and

generating, on the basis of the abstract data structure, a concrete data structure defining a concrete graphical representation in a graphics rendering language using the transformation rules.

22-47. (Canceled)

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.